

LAPORAN PRAKTIKUM 7
PEMROGRAMAN LANJUT
APLIKASI BERBASIS TEXT



Oleh

Nama : Rizqillah
NIM : 1957301020
Kelas : TI 2C
Dosen Pembimbing : Musta'inul Abdi, SST., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER
TAHUN 2021

LEMBAR PENGESAHAN

No. Praktikum : 07/PPL/2C/TI/2021
Judul : Aplikasi Berbasis Text
Nama : Rizqillah
NIM / Kelas : 1957301020 / TI 2C
Jurusan : Teknologi Informasi Dan Komputer
Prodi : Teknik Informatika
Tanggal praktikum : 22 April 2021
Tanggal penyerahan : 27 April 2021
Nilai :

Buketrata, 27 April 2021

Dosen Pembimbing,

Musta'inul Abdi, SST., M.Kom.
NIP. 19911030 20190310 1 5

DAFTAR ISI

LEMBAR PENGESAHAN	i
DAFTAR ISI	ii
DAFTAR GAMBAR	iii
DAFTAR TABEL	iv
BAB I	1
1.1 Tujuan	1
1.2 Argument Command-Line dan System Properties	1
1.2.1 Membaca System Properties	6
1.2.2 Menulis System Properties	6
1.3 Membaca Standard Input	9
1.4 Menangani File	11
1.4.1 Membaca Sebuah File	11
1.4.2 Menulis Sebuah File	13
BAB II	17
2.1 Latihan	17
2.1.1 Spasi menjadi Underscore (_)	17
BAB III	20
3.1 Kesimpulan	20
DAFTAR PUSTAKA	21

DAFTAR GAMBAR

Gambar 1. Hasil program menggunakan Arguments.....	2
Gambar 2. getProperty() method dari class System.....	3
Gambar 3. Hasil program System.getProperties().list(System.out).....	4
Gambar 4. Letak File myProperties.txt.....	7
Gambar 5. Hasil program Coba.java.....	8
Gambar 6. Hasil program FavoriteCharacter.java	10
Gambar 7. Hasil program GreetUser.java.....	10
Gambar 8. Hasil program ReadFile.java.....	13
Gambar 9. Hasil program WriteFile.java.....	16
Gambar 10. Tampilan isi file Coba.txt.....	16
Gambar 11. Hasil program UbahFile.java	18
Gambar 12. Hasil dari program UbahFile.java ke file Tujuan.txt	19

DAFTAR TABEL

Tabel 1. Jenis-jenis System Properties.....	5
---	---

BAB I

PENDAHULUAN

1.1 Tujuan

Pembahasan kali ini akan menitikberatkan pada bahasan penggunaan argument command-line. Selebihnya, Anda akan mempelajari mengenai penggunaan streams untuk mendapatkan nilai input dari user pada saat runtime, sekaligus dalam proses manipulasi file.

Setelah menyelesaikan pembahasan ini, Anda diharapkan dapat :

1. Mendapatkan input dari command-line
2. Mengetahui cara untuk memanipulasi properties dari sistem
3. Membaca standard input
4. Membaca dan menulis file

1.2 Argument Command-Line dan System Properties

Argument command-line adalah informasi yang langsung mengikuti nama program pada baris perintah saat dijalankan. Untuk mengakses argument command-line di dalam program Java cukup mudah. dikarenakan disimpan sebagai string dalam array String yang diteruskan ke main().

Argument command-line adalah argumen yang dikirimkan pada saat menjalankan program java. Argumen yang dikirimkan dari konsol dapat diterima di program java dan dapat digunakan sebagai masukan. Jadi, ini memberikan cara yang mudah untuk memeriksa perilaku program untuk nilai-nilai yang berbeda. Argument command-line dapat diisi dan tiap-tiap kata yang dipisahkan oleh spasi memiliki indexnya masing-masing. Oleh karenanya tipe data yang digunakan pada argument adalah String yang memiliki struktur data array.

Pengguna memasukkan argument command-line saat menjalankan aplikasi dan menentukannya setelah nama kelas yang akan dijalankan. Sebagai contoh, untuk meneruskan argument 1 dan 2 kepada program Java bernama Calculate, dapat dituliskan baris berikut pada command prompt

```
java Calculate 1 2
```

Pada contoh berikut ini, data 1 disimpan pada variabel args[0], begitu pula dengan data 2 yang disimpan pada args[1]. Sehingga, tujuan dari deklarasi String args[] sebagai sebuah parameter pada method main() menjadi jelas.

Adapun contoh program sederhana menggunakan argument command-line adalah sebagai berikut :

```
package ArgumentsCommandLine;

public class CobaArgs {
    public static void main(String[] args) {
        for (String s : args) {
            System.out.println(s);
        }
    }
}
```

Adapun argument yang diisi sebagai berikut :

Arguments: 1 2 3 4 5

Hasil :

```
1
2
3
4
5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 1. Hasil program menggunakan Arguments

Dari program diatas dapat disimpulkan bahwa sebuah program java dengan class Coba dan memiliki method main yang memiliki nilai paramater dari arguments dengan tipe data String dan dengan struktur data array. Kemudian pada bagian properties project tersebut, diberi arguments dengan isi. [1, 2, 3, 4, 5]. Tiap-tiap argument tersebut memiliki index yang dipisahkan dengan spasi.

Selain melewati argument menuju method utama, Anda juga dapat memanipulasi system properties dari command-line.

System properties hampir menyamai environment variables, namun tidak memiliki ketergantungan pada spesifikasi platform yang digunakan. Sebuah property secara sederhana berupa pemetaan antara property name dan value yang dimilikinya. Hal ini ditunjukkan pada Java dalam class Properties. Class System menyediakan sebuah method untuk menentukan system properties yang digunakan, method `getProperties` yang menghasilkan sebuah object Properties. Class yang sama juga menyediakan method `getProperty` yang memiliki dua buah bentuk.

<code>public static String getProperty(String key)</code>
Bentuk ini menghasilkan nilai String dari System Properties yang ditunjukkan oleh key yang ditentukan. Jika hasil menunjukkan nilai null, berarti tidak terdapat property dengan key yang ditentukan.
<code>public static String getProperty(String key, String def)</code>
Bentuk ini juga menghasilkan nilai String dari System Properties sesuai key yang ditentukan. Akan menghasilkan nilai <i>def</i> , sebuah nilai default, jika tidak terdapat property dengan key yang sesuai.

Gambar 2. `getProperty()` method dari class `System`

Anda dapat menggunakan argument opsional `-D` pada perintah Java dalam command-line untuk menambahkan property baru.

```
java -D=value
```

Sebagai contoh, untuk mengatur system property dengan nama `user.home` bernilai `philippines`, gunakan perintah berikut :

```
java -Duser.home=philippines
```

Untuk menampilkan daftar system properties yang tersedia pada sistem Anda, gunakan method `getProperties` seperti yang ditunjukkan sebagai berikut :

```
System.getProperties().list(System.out);
```

Hasil :

```
-- listing properties --
java.runtime.name=Java(TM) SE Runtime Environment
sun.boot.library.path=C:\Program Files\Java\jdk1.8.0_60\jre...
java.vm.version=25.60-b23
java.vm.vendor=Oracle Corporation
java.vendor.url=http://java.oracle.com/
path.separator=;
java.vm.name=Java HotSpot(TM) 64-Bit Server VM
file.encoding.pkg=sun.io
user.script=
user.country=ID
sun.java.launcher=SUN_STANDARD
sun.os.patch.level=
java.vm.specification.name=Java Virtual Machine Specification
user.dir=C:\Users\LENOVO\Documents\NetBeansPro...
java.runtime.version=1.8.0_60-b27
java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment
java.endorsed.dirs=C:\Program Files\Java\jdk1.8.0_60\jre...
os.arch=amd64
java.io.tmpdir=C:\Users\LENOVO\AppData\Local\Temp\
line.separator=
```



```

java.vm.specification.vendor=Oracle Corporation
user.variant=
os.name=Windows 10
sun.jnu.encoding=Cp1252
java.library.path=C:\Program Files\Java\jdk1.8.0_60\bin...
java.specification.name=Java Platform API Specification
java.class.version=52.0
sun.management.compiler=HotSpot 64-Bit Tiered Compilers
os.version=10.0
user.home=C:\Users\LENOVO
user.timezone=
java.awt.printerjob=sun.awt.windows.WPrinterJob

file.encoding=UTF-8
java.specification.version=1.8
user.name=LENOVO
java.class.path=C:\Users\LENOVO\Documents\NetBeansPro...
java.vm.specification.version=1.8
sun.arch.data.model=64
java.home=C:\Program Files\Java\jdk1.8.0_60\jre
sun.java.command=ArgumentsCommandLine.Coba 1 2 3 4 5
java.specification.vendor=Oracle Corporation
user.language=en
awt.toolkit=sun.awt.windows.WToolkit
java.vm.info=mixed mode
java.version=1.8.0_60
java.ext.dirs=C:\Program Files\Java\jdk1.8.0_60\jre...
sun.boot.class.path=C:\Program Files\Java\jdk1.8.0_60\jre...
java.vendor=Oracle Corporation
file.separator=\
java.vendor.url.bug=http://bugreport.sun.com/bugreport/
sun.cpu.endian=little
sun.io.unicode.encoding=UnicodeLittle
sun.desktop=windows
sun.cpu.isalist=amd64
BUILD SUCCESSFUL (total time: 0 seconds)

```

Gambar 3. Hasil program `System.getProperties().list(System.out)`

Analisis :

Dapat dilihat pada sintaks yang telah dijalankan diatas bahwa hasil dari `System.getProperties().list(System.out)`; adalah akan menampilkan seluruh isi properties dari sistem java yang dipakai, beserta informasi lainnya, seperti informasi mengenai sistem operasi yang digunakan dan adapun informasi dari nilai argument command line yang diisi.

Dalam System Properties, kita memeriksa cara aplikasi menggunakan objek Properties untuk mempertahankan konfigurasinya. Platform Java sendiri menggunakan objek Properties untuk mempertahankan konfigurasinya sendiri. Class Sistem memiliki objek Properties yang menjelaskan konfigurasi lingkungan kerja. System Properties mencakup informasi tentang pengguna saat ini, versi runtime Java saat ini, dan karakter yang digunakan untuk memisahkan komponen dari nama jalur file.

Tabel berikut menjelaskan beberapa System Properties yang paling penting :

Tabel 1. Jenis-jenis System Properties

Key	Artinya
"file.separator"	Karakter yang memisahkan komponen jalur sebuah file. "/" adalah contoh pemisah di UNIX dan "\" adalah pemisah di Windows.
"java.class.path"	Path yang digunakan untuk mencari direktori dan arsip JAR yang berisi file class. Elemen jalur class dipisahkan oleh karakter khusus platform yang ditentukan di properti path.separator.
"java.home"	Direktori instalasi untuk Java Runtime Environment (JRE)
"java.vendor"	Nama vendor dari JRE
"java.vendor.url"	Url dari vendor JRE
"java.version"	Nomor versi dari JRE
"line.separator"	Urutan yang digunakan oleh sistem operasi untuk memisahkan baris dalam file teks
"os.arch"	Arsitektur sistem operasi
"os.name"	Nama sistem operasi
"os.version"	Versi sistem operasi
"path.separator"	Karakter pemisah jalur yang digunakan dalam java.class.path
"user.dir"	Direktori kerja pengguna
"user.home"	Direktori home pengguna
"user.name"	Direktori akun pengguna

1.2.1 Membaca System Properties

Sistem Class memiliki dua method yang digunakan untuk membaca properti sistem yaitu : `getProperty` dan `getProperties`.

Sistem Class memiliki dua versi `getProperty` yang berbeda. Keduanya mengambil nilai properti yang disebutkan dalam daftar argumen. Yang lebih sederhana dari dua method `getProperty` mengambil satu argumen, kunci properti Misalnya, untuk mendapatkan nilai `path.separator`, gunakan pernyataan berikut:

```
System.getProperty("path.separator");
```

Method `getProperty` mengembalikan string yang berisi nilai properti. Jika properti tidak ada, maka `getProperty` akan mengembalikan `null`.

Versi lain dari `getProperty` memerlukan dua argumen String: argumen pertama adalah kunci yang harus dicari dan argumen kedua adalah nilai default yang akan dikembalikan jika kunci tidak dapat ditemukan atau jika tidak memiliki nilai. Misalnya, pemanggilan `getProperty` berikut mencari properti Sistem yang disebut `subliminal.message`. Ini bukan properti sistem yang valid, jadi alih-alih mengembalikan nol, metode ini mengembalikan nilai default yang diberikan sebagai argumen kedua: "Properti tidak ada!"

```
System.getProperty("subliminal.message", "Properti tidak ada!");
```

Method terakhir yang disediakan oleh sistem class untuk mengakses nilai properti adalah method `getProperties`, yang mengembalikan objek `Properties`. Objek ini berisi satu set lengkap definisi properti sistem.

1.2.2 Menulis System Properties

Untuk mengubah set properti sistem yang ada, gunakan `System.setProperties`. Method ini mengambil objek `Properties` yang telah diinisialisasi berisi properti yang akan disetel. Method ini menggantikan seluruh set properti sistem dengan set baru yang diwakili oleh objek `Properties`.

Perlu diperhatikan bahwa mengubah properti sistem berpotensi berbahaya dan harus dilakukan dengan cara yang bijak. Banyak properti sistem tidak dibaca ulang setelah start-up dan informasi dari sistem akan berakibat fatal. Mengubah beberapa properti mungkin memiliki efek samping yang tidak terduga.

Contoh berikutnya, PropertiesTest, membuat objek Properties dan menginisialisasinya dengan nama myProperties.txt yang diisi nilai :

subliminal.message = Data sudah ada!

PropertiesTest kemudian menggunakan System.setProperties untuk menginstal objek Properties baru sebagai set properti sistem saat ini.

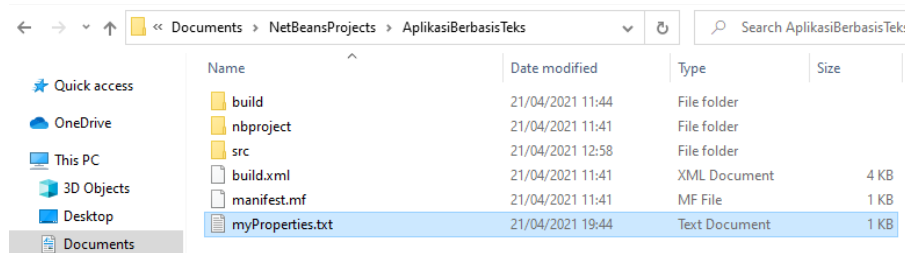
Program :

```
package ArgumentsCommandLine;
import java.io.FileInputStream;
import java.util.Properties;

public class Coba {
    public static void main(String[] args) throws Exception {
        // set up new properties object
        // from file "myProperties.txt"
        FileInputStream propFile = new
        FileInputStream("myProperties.txt");
        Properties p = new Properties(System.getProperties());
        p.load(propFile);

        // set the system properties
        System.setProperties(p);
        // display new properties
        System.getProperties().list(System.out);
    }
}
```

Letak File myProperties.txt :



Gambar 4. Letak File myProperties.txt

File tersebut diletakkan didalam folder project, dan diluar dari folder letak file program yang akan dijalankan.

Hasil :

```
java.vm.specification.vendor=Oracle Corporation
user.variant=
os.name=Windows 10
sun.jnu.encoding=Cp1252
subliminal.message=Data sudah ada!
java.library.path=C:\Program Files\Java\jdk1.8.0_60\bin...
java.specification.name=Java Platform API Specification
java.class.version=52.0
sun.management.compiler=HotSpot 64-Bit Tiered Compilers
os.version=10.0
```

Gambar 5. Hasil program Coba.java

Pada program diatas dapat dilihat bahwa file yang telah dibuat sebelumnya akan mengisi data didalam system properties beserta nama property dan value yang dimiliki. Yaitu dengan nama “subliminal.message=Data sudah ada!”.

Perhatikan bagaimana PropertiesTest membuat objek Properties dengan nama p, yang digunakan sebagai argumen untuk setProperties:

```
Properties p = new Properties(System.getProperties());
```

Pernyataan ini menginisialisasi objek properti baru, p, dengan kumpulan properti sistem saat ini, yang dalam kasus aplikasi kecil ini, adalah kumpulan properti yang diinisialisasi oleh sistem runtime. Kemudian aplikasi memuat properti tambahan ke dalam p dari file myProperties.txt dan menyetel properti sistem ke p. Ini memiliki efek menambahkan properti yang tercantum di myProperties.txt ke kumpulan properti yang dibuat oleh sistem runtime saat memulai. Perhatikan bahwa aplikasi dapat membuat p tanpa objek Properties default, seperti berikut ini:

```
Properties p = new Properties();
```

Perhatikan juga bahwa nilai properti sistem dapat ditimpa! Misalnya, jika myProperties.txt berisi baris berikut, yaitu properti sistem java.vendor akan ditimpa :

```
java.vendor=Acme Software Company
```

Secara umum, berhati-hatilah untuk tidak menimpa properti sistem.

Metode setProperties mengubah kumpulan properti sistem untuk aplikasi yang sedang berjalan. Perubahan ini tidak terus-menerus. Artinya, mengubah properti sistem dalam aplikasi tidak akan memengaruhi pemanggilan interpreter Java di masa mendatang untuk aplikasi ini atau aplikasi lainnya. Sistem runtime menginisialisasi ulang properti sistem setiap kali dimulai. Jika perubahan pada properti sistem bersifat persisten, maka aplikasi harus menulis nilai ke beberapa file sebelum keluar dan membacanya lagi saat startup.

1.3 Membaca Standard Input

Dibandingkan dengan mendapatkan masukan user dari command-line, sebagian user lebih memilih untuk memasukkan data bilamana diminta oleh program pada saat eksekusi. Satu cara dalam melakukan hal ini adalah dengan menggunakan stream. Sebuah stream adalah abstraksi dari sebuah file atau sebuah perangkat yang memungkinkan beberapa set item untuk dibaca atau ditulis. Streams terhubung dengan physical devices seperti keyboards, consoles dan files. Terdapat dua bentuk umum dari streams, byte streams dan character streams. Byte streams digunakan pada data biner, sedangkan character streams digunakan pada karakter Unicode. System.in dan System.out adalah dua contoh dari byte streams yang digunakan dalam Java. Contoh pertama mereferensikan pada keyboard, kemudian contoh kedua mereferensikan pada console.

Untuk membaca karakter dari keyboard, Anda dapat menggunakan byte stream System.in yang terdapat pada object BufferedReader. Baris berikut menunjukkan bagaimana untuk melakukan hal tersebut :

```
BufferedReader br = new BufferedReader(new  
    InputStreamReader(System.in));
```

BufferedReader merupakan salah satu contoh class yang paling banyak digunakan dalam pemograman Java. Fungsi dasar dari BufferedReader adalah membaca file dari input stream. Lebih spesifik lagi bahwa class ini digunakan sebagai buffer dari karakter-karakter dengan tujuan membuat penanganan yang lebih efisien untuk string, character, dan array.

Terdapat dua constructor untuk class BufferedReader ini, yaitu :

- BufferedReader(Reader in)
- BufferedReader(Reader in, int sz)

Method read dari object BufferedReader selanjutnya digunakan untuk membaca nilai input dari perangkat input.

```
ch=(int)br.read(); //method read menghasilkan nilai integer
```

Cobalah contoh kode berikut :

```
package ArgumentsCommandLine;  
import java.io.*;  
  
class FavoriteCharacter {  
    public static void main(String args[]) throws IOException {  
        System.out.println("Hi, what's your favorite  
character?");  
        char favChar;
```

```

        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        favChar = (char) br.read();
        System.out.println(favChar + " is a good choice!");
    }
}

```

Hasil :

```

Hi, what's your favorite character?
Naruto
N is a good choice!
BUILD SUCCESSFUL (total time: 7 seconds)

```

Gambar 6. Hasil program FavoriteCharacter.java

Jika Anda lebih memilih untuk membaca keseluruhan baris daripada membaca satu karakter tiap waktu, gunakan method `readLine` :

```
str = br.readLine();
```

Berikut ini sebuah program yang hampir menyerupai contoh sebelumnya, namun membaca keseluruhan string, bukan satu karakter.

```

package ArgumentsCommandLine;
import java.io.*;

class GreetUser {
    public static void main(String args[]) throws IOException {
        System.out.println("Hi, what's your name?");
        String name;
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        name = br.readLine();
        System.out.println("Nice to meet you, " + name + "! :)");
    }
}

```

Hasil :

```

Hi, what's your name?
Rizqillah
Nice to meet you, Rizqillah! :)
BUILD SUCCESSFUL (total time: 5 seconds)

```

Gambar 7. Hasil program GreetUser.java

Pada saat menggunakan streams, jangan lupa untuk mengimport package `java.io` seperti yang ditunjukkan dibawah ini :

```
import java.io.*;
```

Satu hal lagi yang perlu untuk diingat, pembacaan dari streams dapat menyebabkan terjadinya exception. Jangan lupa untuk menangani exception tersebut menggunakan perintah `try-catch` atau dengan mengindikasikan exception pada klausa `throws` dalam method.

1.4 Menangani File

Pada beberapa kasus, masukan data disimpan pada sebuah file. Selanjutnya, terdapat beberapa cara jika Anda ingin menyimpan output dari program pada sebuah file. Pada sistem terkomputerisasi, data dari Siswa yang dapat digunakan sebagai input oleh sistem umumnya terseimpan pada sebuah file terpisah. Kemudian, salah satu kemungkinan output dari sistem adalah informasi tentang mata pelajaran yang diikuti oleh siswa. Sekali lagi, output dalam hal ini dapat disimpan dalam sebuah file. Seperti yang terlihat pada aplikasi, terdapat suatu kebutuhan untuk membaca dan menulis sebuah file. Anda akan mempelajari tentang file input dan output pada bagian ini.

1.4.1 Membaca Sebuah File

Untuk proses input dan output pada program Java, package `java.io` memiliki class-class yang sangat lengkap untuk mendukung tujuan ini. Setiap streams ini bisa merepresentasikan sumber-sumber dari input dan juga target dari output. Package `java.io` juga mendukung banyak data, seperti data primitif, objek, dan sebagainya.

Secara umum stream bisa diartikan sebagai urutan dari data. Terdapat 2 jenis stream, yaitu:

- `InputStream` yang digunakan untuk membaca data dari sumbernya
- `OutputStream` yang digunakan untuk menulis data pada tujuan atau target

Untuk file dan jaringan (*networks*), Java memberikan dukungan fleksibel dan juga kuat. Namun, kita hanya akan melihat dasarnya saja melalui contoh-contoh yang sering digunakan secara umum.

Class `FileInputStream` pada Java memperoleh byte input dari file. File apa yang tersedia tergantung pada lingkungan host. Ini digunakan untuk membaca data berorientasi byte (streams byte mentah) seperti data gambar, audio, video dll. dan juga dapat membaca data streams karakter. Namun, untuk membaca aliran karakter, disarankan untuk menggunakan Class `FileReader`.

Adapun method yang tersedia dalam class `FileInputStream` yaitu sebagai berikut :

- **`Available()`**, Mengembalikan perkiraan jumlah byte tersisa yang dapat dibaca (atau dilewati) dari aliran input tanpa pemblokiran oleh pemanggilan metode berikutnya untuk aliran input.
- **`Close()`**, Tutup `InputStream` dan lepaskan semua sumber daya sistem yang terkait dengan stream.

- **Finalize()**, Memastikan bahwa method close InputStream dipanggil saat tidak ada lagi referensi.
- **getChannel()**, Mengembalikan objek FileChannel unik yang terkait dengan InputStream.
- **getFd()**, Mengembalikan objek FileDescriptor yang mewakili koneksi ke file sebenarnya dalam sistem file yang digunakan oleh InputStream.
- **Read()**, Membaca byte data dari InputStream.
- **Read(byte[] b)**, Membaca hingga b.length byte data dari InputStream ke dalam array byte.
- **Read(byte[] b, int off, int len)**, Membaca hingga len byte data dari InputStream ke dalam array byte.
- **Skip(long n)**, Melompati dan membuang n byte data dari InputStream

Untuk membaca sebuah file, Anda dapat menggunakan class FileInputStream. Berikut ini adalah salah satu constructor dari class tersebut :

```
FileInputStream(String filename)
```

Constructor tersebut membuat sebuah koneksi terhadap file dimana nama dari file tersebut ditunjukkan sebagai sebuah argument. Exception berupa FileNotFoundException akan muncul jika file tidak ditemukan atau tidak dapat dibuka dan kemudian dibaca.

Setelah membuat sebuah input stream, Anda kemudian dapat menggunakannya untuk membaca sebuah file dengan menggunakan method read. Method read menghasilkan sebuah nilai integer, dan akan menunjukkan nilai 1 jika telah mencapai batas akhir file.

Berikut ini contoh program :

```
package ArgumentsCommandLine;
import java.io.*;

class ReadFile {
    public static void main(String args[]) throws IOException {
        System.out.println("What is the name of the file to read from?");
        String filename;
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        filename = br.readLine();
        System.out.println("Now reading from " + filename + "...");
        FileInputStream fis = null;
        try {
```

```

        fis = new FileInputStream(filename);
    } catch (FileNotFoundException ex) {
        System.out.println("File not found.");
    }
    try {
        char data;
        int temp;
        do {
            temp = fis.read();
            data = (char) temp;
            if (temp != -1) {
                System.out.print(data);
            }
        } while (temp != -1);
    } catch (IOException ex) {
        System.out.println("Problem in reading from the
file.");
    }
}
}
}

```

Hasil :

```

What is the name of the file to read from?
myProperties.txt
Now reading from myProperties.txt...
subliminal.message = Data sudah ada!
BUILD SUCCESSFUL (total time: 7 seconds)

```

Gambar 8. Hasil program ReadFile.java

Program diatas mencoba membaca sebuah file dengan meminta inputan dari user untuk menginput nama file yang akan dibaca, dikarenakan sebelumnya telah membuat file dengan nama myProperties.txt yang berisi data “subliminal.message = Data sudah ada!” oleh karenanya, ketika pembacaan file selesai akan muncul data yang sebelumnya telah diisi didalam file myProperties.txt.

1.4.2 Menulis Sebuah File

File dapat dibaca menggunakan Reader atau Stream di java. Pembaca bagus digunakan untuk data teks tetapi untuk bekerja dengan data biner harus menggunakan Stream. FileInputStream digunakan untuk membuka stream untuk membaca data dari file. Disini akan membuat suatu cara untuk menulis sebuah file menggunakan FileOutputStream.

FileOutputStream adalah OutputStream untuk menulis data ke File atau ke FileDescriptor. Apakah file tersedia atau mungkin dibuat tergantung pada platform yang mendasarinya. Beberapa platform, khususnya, mengizinkan file

dibuka untuk ditulis hanya dengan satu `FileOutputStream` (atau objek penulisan file lainnya) pada satu waktu. Dalam situasi seperti itu, konstruktor di class ini akan gagal jika file yang terlibat sudah terbuka.

`FileOutputStream` dimaksudkan untuk menulis aliran byte mentah seperti data gambar. Jika akan menulis stream karakter, pertimbangkan untuk menggunakan `FileWriter`.

Untuk menuliskan sebuah file, dapat menggunakan class `FileOutputStream`. Berikut ini salah satu constructor yang dapat digunakan.

```
FileOutputStream(String filename)
```

Constructor tersebut menyediakan jalur output stream terhadap sebuah file yang akan ditulis. Sebuah Exception berupa `FileNotFoundException` akan muncul jika file yang dimaksud tidak dapat dibuka untuk ditulis.

Jika output stream telah dibuat, Anda dapat menggunakannya untuk menulis file yang dituju menggunakan method `write`. Method tersebut menggunakan penandaan sebagai berikut :

```
void write(int b)
```

Parameter `b` mereferensikan data yang akan dituliskan pada file sesuai dengan hasil output stream.

Adapun method yang disediakan dalam class `FileOutputStream` sebagai berikut :

- **`Close()`**, Menutup `FileOutputStream` dan melepaskan semua sumber daya sistem yang terkait dengan stream.
- **`Finalize()`**, Memastikan bahwa method `close` `OutputStream` dipanggil saat tidak ada lagi referensi.
- **`getChannel()`**, Mengembalikan objek `FileChannel` unik yang terkait dengan `InputStream`.
- **`getFd()`**, Mengembalikan `FileDescriptor` yang terkait dengan stream tersebut
- **`write(byte[] b)`**, Menulis `b.length` byte dari array byte yang ditentukan ke `FileOutputStream`.
- **`write(byte[] b, int off, int len)`**, Menulis `len` byte dari array byte yang ditentukan mulai dari `offset` ke `FileOutputStream` tersebut.
- **`write(int b)`**, Menulis byte yang ditentukan ke `FileOutputStream`.

Program berikut menunjukkan contoh penulisan terhadap file :

```
package ArgumentsCommandLine;
import java.io.*;

class WriteFile {
    public static void main(String args[]) throws IOException {
        System.out.println("What is the name of the file to be
writtento ?");
        String filename;
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        filename = br.readLine();
        System.out.println("Enter data to write to " + filename
+ "...");
        System.out.println("Type q$ to end.");
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(filename);
        } catch (FileNotFoundException ex) {
            System.out.println(
                "File cannot be opened for writing.");
        }

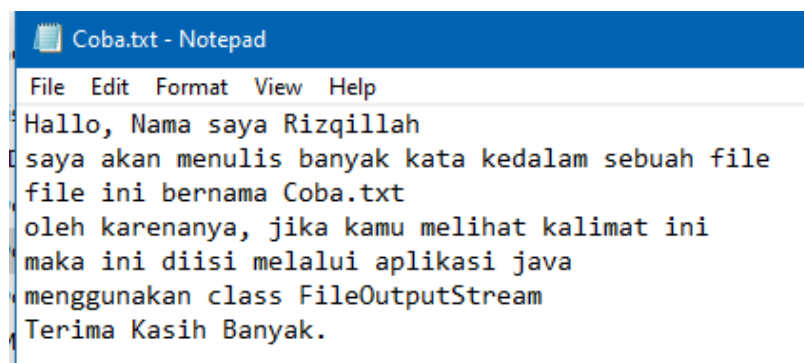
        try {
            boolean done = false;
            int data;
            do {
                data = br.read();
                if ((char) data == 'q') {
                    data = br.read();
                    if ((char) data == '$') {
                        done = true;
                    } else {
                        fos.write('q');
                        fos.write(data);
                    }
                } else {
                    fos.write(data);
                }
            } while (!done);
        } catch (IOException ex) {
            System.out.println("Problem in reading from the
file.");
        }
    }
}
```

Hasil :

```
What is the name of the file to be writtentto ?
Coba.txt
Enter data to write to Coba.txt...
Type q$ to end.
Hallo, Nama saya Rizqillah
saya akan menulis banyak kata kedalam sebuah file
file ini bernama Coba.txt
oleh karenanya, jika kamu melihat kalimat ini
maka ini diisi melalui aplikasi java
menggunakan class FileOutputStream
Terima Kasih Banyak.
q$
BUILD SUCCESSFUL (total time: 1 minute 27 seconds)
```

Gambar 9. Hasil program WriteFile.java

Hasil dalam File Coba.txt :



Gambar 10. Tampilan isi file Coba.txt

Hal tersebut bisa terjadi dikarenakan telah diisi nilai yang meminta inputan nama file yang akan diisi. Dan sebelumnya telah disiapkan file bernama Coba.txt didalam folder project oleh karenanya ketika mengisi nama file tersebut, maka program akan mendapati file tersebut kemudian meminta inputan dari user untuk menginput hal yang diinginkan, dan program akan membaca per-perkata yang dimasukkan oleh user. Dan jika user menginput huruf 'q' maka program akan menganggap user menginput q tersebut kedalam file Coba.txt juga.

Akan tetapi jika user menginput huruf q\$ bersamaan dengan cara menekan enter setelahnya, maka program akan keluar dari perulangan do while yang melakukan perulangan selama nilai done = false. Dan setelah diisi kata-kata disaat pembacaan file berhasil, maka data yang diisi tersebut akan diinput kedalam file yang sebelumnya dipanggil, yaitu file Coba.txt.

BAB II PEMBAHASAN

2.1 Latihan

2.1.1 Spasi menjadi Underscore (_)

Buatlah sebuah program yang memuat dua String sebagai argument, sumber dan nama file tujuan. Kemudian, baca file sumber dan tuliskan isi dari file tersebut terhadap file tujuan, seluruh spasi yang ada (' ') diubah menjadi underscore (' _ ').

Program :

```
package ArgumentsCommandLine;
import java.io.*;

public class UbahFile {
    public UbahFile(String sumber, String tujuan) {
        FileInputStream fis = null;
        try {
            fis = new FileInputStream(sumber);
        } catch (FileNotFoundException ex) {
            System.out.println("File Tidak Ditemukan!");
        }

        try {
            char data;
            int temp;
            FileOutputStream fos = new FileOutputStream(tujuan);
            do {
                temp = fis.read();
                data = (char) temp;
                if (temp != -1) {
                    try {
                        if ((char) temp == ' ') {
                            data = '_';
                            fos.write(data);
                        } else {
                            fos.write(data);
                        }
                    } catch (FileNotFoundException ex) {
                        System.out.println("File tidak dapat
Dibuka dan Ditulis!");
                    }
                }
            } while (temp != -1);
            System.out.println("Berhasil menyimpan pada file " +
tujuan);
        }
    }
}
```

```

        } catch (IOException ex) {
            System.out.println("Masalah terjadi selama proses
membaca!");
        }
    }

    public static void main(String args[]) throws IOException {
        System.out.println("Nama File sumber beserta Extensi ?");
        String sumber;
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        sumber = br.readLine();
        System.out.println("Mencoba membaca File " + sumber +
"...");
        System.out.println("Nama File tujuan yang akan disimpan
?");
        String tujuan;
        BufferedReader bd = new BufferedReader(new
InputStreamReader(System.in));
        tujuan = bd.readLine();
        System.out.println("Mencoba Menulis pada File " + tujuan
+ "...");
        new UbahFile(sumber, tujuan);
    }
}

```

Hasil :

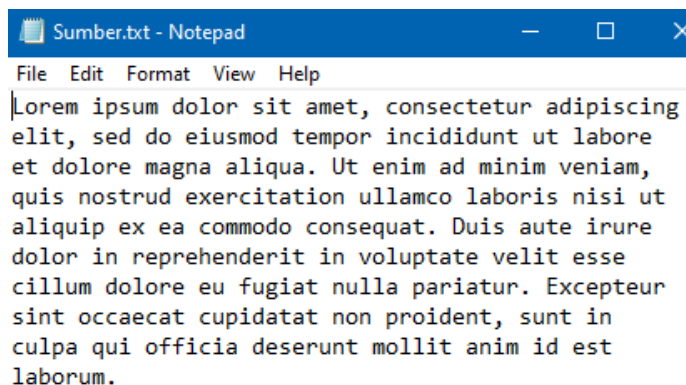
```

Nama File sumber beserta Extensi ?
Sumber.txt
Mencoba membaca File Sumber.txt...
Nama File tujuan yang akan disimpan ?
Tujuan.txt
Mencoba Menulis pada File Tujuan.txt...
Berhasil menyimpan pada file Tujuan.txt
BUILD SUCCESSFUL (total time: 16 seconds)

```

Gambar 11. Hasil program UbahFile.java

File Sumber.txt :

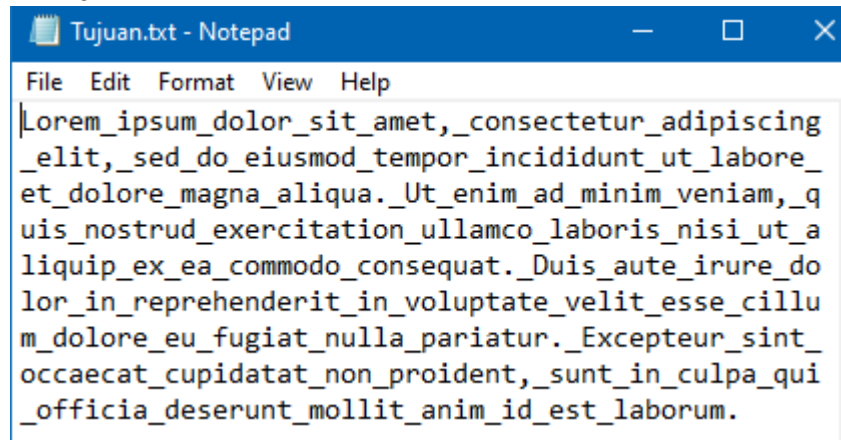


```

File Edit Format View Help
Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est
laborum.

```

Hasil File Tujuan.txt :



```
File Edit Format View Help
Lorem_ipsum_dolor_sit_amet,_consectetur_adipiscing
_elit,_sed_do_eiusmod_tempor_incididunt_ut_labore_
et_dolore_magna_aliqua._Ut_enim_ad_minim_veniam,_q
uis_nostrud_exercitation_ullamco_laboris_nisi_ut_a
liquip_ex_ea_commodo_consequat._Duis_aute_irure_do
lor_in_reprehenderit_in_voluptate_velit_esse_cillu
m_dolore_eu_fugiat_nulla_pariatur._Excepteur_sint_
occaecat_cupidatat_non_proident,_sunt_in_culpa_qui
_officia_deserunt_mollit_anim_id_est_laborum.
```

Gambar 12. Hasil dari program UbahFile.java ke file Tujuan.txt

Analisis :

Pada program diatas telah dilakukan pemindahan data dari file Sumber.txt kedalam file Tujuan.txt. akan tetapi pemindahan yang dilakukan adalah bukan pemindahan melalui pengcopy-an file. Melainkan melalui pembacaan tiap-tiap kata yang ada, kemudian melakukan pengecekan terhadap kata yang diambil sebelum diinput kedalam file Tujuan.txt, yaitu mengecek apakah kata yang diinput adalah spasi (" "). Jika iya, maka akan mengganti spasi tersebut dengan underscore (_). Kemudian diisi kedalam file Tujuan.txt.

Hal tersebut akan dilakukan berulang kali selama nilai dari data sumber belum mencapai nilai -1. Jika data mencapai -1, maka akan keluar dari perulangan do while. Dan data dalam file Tujuan.txt yang sebelumnya kosong akan terisi data dari file Sumber.txt yang telah mengubah spasi menjadi underscore.

BAB III PENUTUP

3.1 Kesimpulan

Dapat disimpulkan bahwa dalam program java telah disediakan sebuah Arguments Command-Line yang bisa diisi dengan nilai apapun kedalamnya. Arguments tersebut bisa diisi ketika akan menjalankan file java jika dijalankan melalui CLI(Command Line Interface), dengan cara mengisi nilai setelah pemanggilan nama file. Contohnya seperti :

```
java Coba Arguments
atau
java Coba ini adalah baris untuk arguments command line
```

Akan tetapi, jika menggunakan platform IDE seperti NetBeans, maka pengisian arguments dilakukan pada properti project dengan cara klik kanan pada project dan memilih properti. Kemudian ke bagian run, nantinya akan ada form input untuk mengisi argument yang diinginkan.

Dalam System Properties, dapat digunakan untuk memeriksa cara aplikasi menggunakan objek Properties untuk mempertahankan konfigurasinya. Platform Java sendiri menggunakan objek Properties untuk mempertahankan konfigurasinya sendiri. Class System memiliki objek Properties yang menjelaskan konfigurasi lingkungan kerja. System Properties mencakup informasi tentang pengguna saat ini, versi runtime Java saat ini, dan karakter yang digunakan untuk memisahkan komponen dari nama jalur file.

Class FileInputStream pada Java memperoleh byte input dari file. File apa yang tersedia tergantung pada lingkungan host. Ini digunakan untuk membaca data berorientasi byte (streams byte mentah) seperti data gambar, audio, video dll. dan juga dapat membaca data streams karakter. Namun, untuk membaca aliran karakter, disarankan untuk menggunakan Class FileReader.

FileOutputStream adalah Stream untuk menulis data ke File atau ke FileDescriptor. Apakah file tersedia atau mungkin dibuat tergantung pada platform yang mendasarinya. Beberapa platform, khususnya, mengizinkan file dibuka untuk ditulis hanya dengan satu FileOutputStream (atau objek penulisan file lainnya) pada satu waktu. Dalam situasi seperti itu, konstruktor di class ini akan gagal jika file yang terlibat sudah terbuka.

DAFTAR PUSTAKA

- [1] Govinda Sai. 07 Februari 2018. Java Command Line Arguments. <https://www.tutorialspoint.com/Java-command-line-arguments>.
TutorialsPoint.
- [2] JavaTPoint. Java Command Line Arguments. <https://www.javatpoint.com/command-line-argument>.
- [3] Oracle Java Documentation. Command-Line Arguments. <https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html>.
- [4] Oracle Java Documentation. System Properties. <https://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html>.
- [5] Bahasa Java. Membaca Input Dari User Menggunakan Class BufferedReader pada Java. <http://bahasajava.com/membaca-input-user-menggunakan-class-bufferedReader-java/>.
- [6] Bahasa Java. Memahami Stream pada Program Java. <http://bahasajava.com/memahami-stream-pada-program-java/>.
- [7] JavaTPoint. Java FileInputStream Class. <https://www.javatpoint.com/java-fileinputstream-class>.
- [8] Oracle Java Documentation. Class FileInputStream. <https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html>.
- [9] Oracle Java Documentation. Class FileOutputStream. <https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html>.
- [10] Pankaj. Java Input Stream to File. <https://www.journaldev.com/918/java-inputstream-to-file-example>. JournalDev.
- [11] Oracle Java Documentation. Class System. <https://docs.oracle.com/javase/7/docs/api/java/lang/System.html>.